MULTIPLE-VALUED PLA MINIMIZATION BY CONCURRENT MULTIPLE AND MIXED SIMULATED ANNEALING*

Cem Yildirim and Jon T. Butler
Dept. of Electr. and Comp. Eng.
Naval Postgraduate School
Monterey, CA 93943-5004 U.S.A.

Chyan Yang
Inst. of Mangmnt Sci. & Inst. of Info. Mangmnt
National Chiao Tung University
Hsinchu, TAIWAN, R.O.C.

Abstract

We analyze simulated annealing applied to multiple-valued programmable logic array (MVL PLA) design. Of specific interest is the use of parallel processors. We consider the use of loosely-coupled, coarse-grained parallel systems, and study the relationship between the quality of the solution and computation time, on the one hand, and simulated annealing parameters, start temperature, cooling rate, etc., on the other. We also investigate simulated annealing where there is a mixture of move types. The mixed move approach provides improvement in both the number of product terms and computation time.

1: Introduction

Simulated annealing is a technique for obtaining approximate solutions to combinatorial optimization problems. Advantages include ease of implementation and the potential to find minimal solutions [12]. Essentially, it is a search of the solution space with the goal of finding a solution of minimum cost. Three steps are repeatedly applied [12].

1) Create a new solution from the current solution.
2) Calculate the cost of the new solution.
3) If the change in cost of the new solution is below some threshold, make it the current solution.

Simulated annealing is well-suited to the MVL PLA minimization problem. Given an expression in the form of a set of product terms, our algorithm [2] divides and recombines the product terms, gradually progressing toward a solution with fewer product terms. Unlike other minimization techniques (classified as direct-cover methods), this technique manipulates product terms directly, breaking them up and joining them in different ways while reducing the total number of product terms. Manipulation of product terms is done nondeterministically. That is, randomly chosen product terms are randomly combined (cost decreasing move), reshaped or divided (cost increasing move). Although cost increasing moves take the solution away from the optimal solution, they allow escape from local minima. Since the solution space in MVL PLA minimization is very large, there are usually a large number of moves before a minimal or near-minimal solution is achieved. To direct the search toward improved solutions, a bias is applied to the probability of acceptance of a cost increasing move. That is, initially the probability of accepting such a move is high, between 0.5 and 1.0. However, as the time passes, this probability decreases. The result is that, at first, wide excursions are made in the solution space, while near the end, only minima are explored. When the probability of accepting a cost increasing move is low, simulated annealing is usually in a minima, and the probability of escape is also low.

2: Background

A product term is expressed as $c\ {}^{a_1}x_1^{b_1}\ {}^{a_2}x_2^{b_2} \ldots {}^{a_n}x_n^{b_n}$, where $c \in \{1,2, \ldots, r\text{-}1\}$ is a nonzero constant, where the literal function is given as

$$
{}^{a_i}x_i^{b_i} \begin{array}{l} = r-1 \;\; \textit{if} \;\; a_i \leq x_i \leq b_i \\ = 0 \;\; \textit{otherwise} \end{array}
$$

and where concatenation is the min function; i.e. $xy = \min(x,y)$. Since the literal function takes on only values 0 or $r\text{-}1$, the product of literals is also 0 or $r\text{-}1$, while the complete term takes on values 0 or c. An r-valued function $f(x_1,x_2,\ldots,x_n)$ takes on values from $\{0,1,\ldots,r\text{-}1\}$ for each assignment of values to the variables, which are also r-valued; i.e. $x_i \in \{0,1,2,\ldots,r\text{-}1\}$. A function can be represented by a sum of these product terms as shown on the next page,

*Research supported by the Naval Research Laboratory, Washington, DC through direct funds at the Naval Postgraduate School, Monterey CA.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **MAY 1993**	2. REPORT TYPE	3. DATES COVERED

4. TITLE AND SUBTITLE **Multiple-Valued PLA Minimization by Concurrent Multiple and Mixed Simulated Annealing**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School,Department of Electrical and Computer Engineering,Monterey,CA,93943**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
We analyze simulated annealing applied to multiple-valued programmable logic array (MVL PLA) design. Of spec@c interest is the use of parallel processors. We consider the use of loosely-coupled, coarsegrainedparallel systems, and study the relationship between the quality of the solution and computation time, on the one hand, and simulated annealing parameters, start temperature, cooling rate, etc., on the other. We also investigate simulated annealing where there is a mixture of move types. The mixed move approach provides improvement in both the number of product terms and computation time.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**7**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

$$\begin{aligned} f(x_1,x_2,\ldots,x_n) = {} & c_1\ {}^{a_{1,1}}x_1^{b_{1,1}}\ {}^{a_{1,2}}x_2^{b_{1,2}} \ldots {}^{a_{1,n}}x_n^{b_{1,n}} \\ & + c_2\ {}^{a_{2,1}}x_1^{b_{2,1}}\ {}^{a_{2,2}}x_2^{b_{2,2}} \ldots {}^{a_{2,n}}x_n^{b_{2,n}} \\ & + c_3\ {}^{a_{3,1}}x_1^{b_{3,1}}\ {}^{a_{3,2}}x_2^{b_{3,2}} \ldots {}^{a_{3,n}}x_n^{b_{3,n}} \\ & + \ldots \quad . \end{aligned} \quad ,$$

where + is truncated sum, i.e. $a+b = \max\{a+b\ ,\ r-1\}$, with the right sum as ordinary addition, where a and b are viewed as integers.

A minimal sum-of-products expression for $f(x_1,x_2,\ldots,x_n)$ is one with the fewest product terms. Our approach to finding such a solution is based on Dueck *et al* [2]. Specifically, given a set of product terms that sum to a given function, we derive another set by making a move. Similarly, a move is made from the next set, etc. until finally a minimal or near-minimal set is formed. As in [2], we investigate two kinds of moves.

1) Cut-or-Combine: Two chosen product terms are combined into one, if possible. If not, one is chosen randomly, with probability 0.5. If the chosen product term is a 1 minterm (i.e. a product term of the form $1\ {}^{a_1}x_1^{b_1}\ {}^{a_2}x_2^{b_2} \ldots {}^{a_n}x_n^{b_n}$, where $a_i=b_i$ for all i), the current move is abandoned and another pair of product terms is chosen. Otherwise, the chosen product term is divided into two. Division occurs either along the logic values or geometrically [2].

2) Reshape: This move, like Cut-or-Combine, operates on two product terms, and combines the pair if a combine is possible. If not, a consensus term is formed. The fewest product terms are chosen to cover all remaining minterms [2].

In this paper, instead of committing to either Cut-or-Combine or Reshape, we use a mixture of both with different paths, and this is performed concurrently in a distributed system. The results of three different approaches are investigated: 1) different paths in the solution space, 2) mixing types of moves, and 3) concurrency.

3: Parallel methods for simulated annealing

3.1: Different paths

One way to improve the final result is to choose different paths. This approach was tested using the Reshape algorithm on the ten different functions. C1,C2,..,C10, used in Dueck *et al* [2]. Each has 200 minterms. For every function, eight different paths were chosen. Table 1 shows the results. Each entry shows two results, the number of product terms achieved and the computation time (in parentheses). The computation time is CPU seconds on a Sun Workstation running the SunOS Release 4.1.1-GFX-Rev2 operating system. The column "OUT" shows the smallest number of product terms and is considered to be the output of the algorithm. In this column, parentheses enclose the total computation time over the eight paths. Since this was performed on one processor, this total is the computation time for this (sequential) version of the algorithm. The average of the product terms and computation times for each path are given in the last row. Table 1 shows a clear dependence on the path. For example, the eight paths on function C4 yield five different values for a near-minimal solution, 80, 81, 82, 83 and 86 product terms.

This experiment shows that a slight improvement in the average is achieved by performing eight rather than one simulated annealing experiment. Taking the best from eight yields an average of 82.4 product terms over the ten functions. We can represent the performance of one simulated annealing experiment by averaging the averages for each path in Table 1, yielding 84.2. This improvement requires a large computation time, an average total of 470.5 seconds vs. 58.0 seconds for one path (calculated by averaging the times for each path). Such a large computation time can be reduced significantly by using multiple processors. Because there is no need to exchange information, the various paths can be executed concurrently. The concurrent execution is discussed in Section 3.3 .

3.2: Annealing with mixed moves

It was shown in Dueck *et al* [2] that Reshape produces overall better results than Cut-or-Combine. However, this was not true of every function tried; Cut-or-Combine did better in a few cases. With this experience, we investigated the use of simulated annealing in which Cut-or-Combine moves were mixed with Reshape moves (Fig. 1). Different mixture ratios were tried, all of which used Reshape more than Cut-or-Combine. The best results were achieved when Cut-or-Combine moves occurred 4% of the time and Reshape moves occurred 96% of the time. Our results are shown in Table 2 (with 4% / 96% mixture ratio). This experiment yields an average number of product terms of 81.7 compared with 84.2 for a single experiment using Reshape without Cut-or-Combine.

3.3: Concurrency in simulated annealing with multiple and mixed moves

We have shown that the use of more than one path with and without mixed moves gives a reduction in the number of product terms over the use of one path. However, substantial computation time is required when one processor is used. A speedup can be achieved by using

Initial Temp	**: 0.7**
Max Valid Factor	**: 4**
Max Frozen	**: 5**
Cooling Rate	**: 0.93**
Max Try Factor	**: 25**

FNC	P0	P1	P2	P3	P4	P5	P6	P7	RSAV	OUT
C1	88(59)	86(53)	87(61)	87(56)	87(57)	86(60)	88(56)	90(57)	87.3(57.4)	86(459)
C2	85(63)	86(66)	85(58)	86(57)	86(56)	84(58)	85(63)	86(62)	85.3(60.4)	84(483)
C3	88(59)	89(67)	91(79)	90(58)	89(56)	90(55)	88(65)	89(55)	89.2(61.7)	88(494)
C4	82(60)	83(52)	83(51)	86(52)	80(73)	82(48)	81(54)	82(53)	82.8(55.4)	80(443)
C5	81(51)	82(62)	81(57)	80(62)	79(58)	82(55)	80(52)	80(65)	80.6(57.7)	79(462)
C6	83(54)	81(56)	86(52)	83(55)	82(57)	83(50)	86(61)	82(54)	83.2(54.9)	81(439)
C7	88(57)	88(68)	87(61)	89(57)	87(68)	90(54)	89(58)	87(57)	84.8(60.0)	87(548)
C8	79(54)	79(54)	78(51)	77(48)	81(60)	81(46)	79(49)	79(49)	79.1(51.4)	77(411)
C9	81(55)	83(57)	82(66)	83(56)	83(63)	82(65)	82(56)	83(73)	82.4(61.4)	81(491)
C10	84(64)	84(55)	84(56)	83(60)	81(63)	87(60)	84(63)	83(54)	83.8(59.4)	81(475)
PHAV	83.9(57.6)	84.1(59.0)	84.4(59.2)	84.4(56.1)	83.5(61.1)	84.7(55.1)	84.2(57.7)	84.1(57.9)	84.2(58.0)	**82.4(470.5)**

Table 1. Sequential multiple paths with regular Reshape.

more than one processor in a distributed system. To investigate this, we used eight Sun Workstations. Fig. 2 shows a program for this distributed system. The host begins by sending the name of the file (in which the input function is placed) to the other processors and later assigns itself as processor 0. Each processor chooses the paths randomly. The probability of two processors picking identical next solution states is reduced by assigning different seeds for the random generators of the processors. The probability that two or more processors choose the same pair of product terms can be calculated as $1-\frac{M!}{(M-N)!\,M^N}$, where $M = \binom{P}{2}$, N is the number of processors used, and P is the number of product terms, in the function to be minimized. For instance, with 8 processors and a function with 200 product terms this probability is 0.0014. The probability of going to the same next solution state is even less. Because of this small probability, the program does not check if two processors have started with the same next solution state, nor if, at any point in the computation, the solution is the same. In this way, communication is required only at the beginning and at the end of the process.

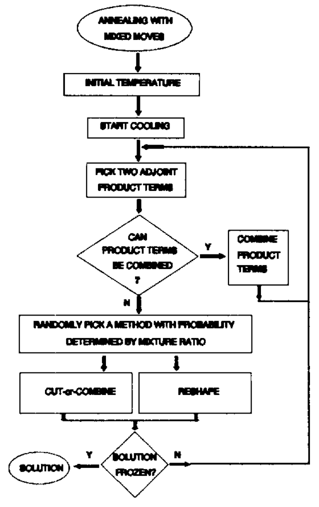
ANNEALING WITH MIXED MOVES
INITIAL TEMPERATURE
START COOLING
PICK TWO ADJOINT PRODUCT TERMS
CAN PRODUCT TERMS BE COMBINED ?
Y
COMBINE PRODUCT TERMS
N
RANDOMLY PICK A METHOD WITH PROBABILITY DETERMINED BY MIXTURE RATIO
CUT-or-COMBINE
RESHAPE
SOLUTION FROZEN?
Y
SOLUTION
N

Figure 1. Simulated annealing with mixed moves.

Initial Temp	**: 0.6**
Max Valid Factor	**: 4**
Max Frozen	**: 5**
Cooling Rate	**: 0.94**
Max Try Factor	**: 25**
Cut-or-Combine/Resh Ratio	**: 4%/96%**

FNC	RAVG	P0	P1	P2	P3	P4	P5	P6	P7	OUT
C1	87.3(57.4)	88(57)	87(70)	84(89)	86(70)	85(78)	87(63)	89(93)	90(66)	84(562)
C2	85.3(60.4)	83(66)	85(88)	83(82)	86(91)	84(78)	86(73)	85(96)	85(76)	83(651)
C3	89.2(61.7)	88(97)	89(96)	89(94)	89(77)	89(67)	87(79)	89(79)	89(83)	87(672)
C4	82.8(55.4)	82(65)	82(63)	81(49)	78(53)	83(53)	80(58)	85(62)	81(61)	78(491)
C5	80.6(57.7)	80(91)	79(77)	82(77)	81(67)	82(77)	82(92)	82(73)	80(61)	79(577)
C6	83.2(54.9)	83(70)	82(63)	83(61)	82(96)	86(67)	84(58)	83(61)	84(59)	82(562)
C7	84.8(60.0)	87(61)	86(84)	87(67)	88(99)	86(67)	91(80)	88(89)	88(108)	86(671)
C8	79.1(51.4)	78(67)	79(76)	79(64)	78(61)	78(55)	77(69)	76(61)	79(70)	76(602)
C9	82.4(61.4)	80(87)	84(88)	83(90)	80(67)	83(63)	84(67)	83(65)	83(78)	80(601)
C10	83.8(59.4)	82(73)	86(83)	85(63)	84(75)	83(63)	85(71)	82(73)	84(86)	82(596)
AV	84.2(58.0)	83.1(73.4)	83.9(78.8)	83.6(73.6)	83.2(75.6)	83.9(66.8)	84.3(71.0)	84.3(75.2)	84.3(74.8)	**81.7(598.5)**

Table 2. Sequential multiple paths with mixed annealing.

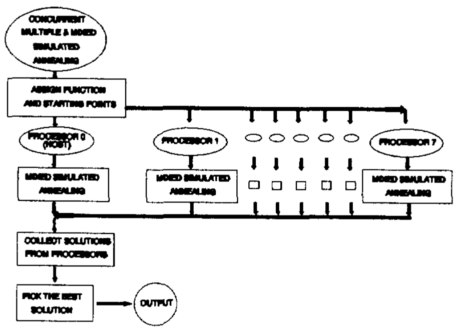
CONCURRENT MULTIPLE & MIXED SIMULATED ANNEALING
ASSIGN FUNCTION AND STARTING POINTS
PROCESSOR 0 (HOST)
PROCESSOR 1
PROCESSOR 7
MIXED SIMULATED ANNEALING
MIXED SIMULATED ANNEALING
MIXED SIMULATED ANNEALING
COLLECT SOLUTIONS FROM PROCESSORS
PICK THE BEST SOLUTION
OUTPUT

Figure 2. Concurrent implementation of multiple path with mixed moves.

	FUNCTION 1			FUNCTION 2		
HEURISTIC	IN	OUT	Time(SEC)	IN	OUT	Time (SEC)
CUT-or-COMBINE	14	19	673	5	4	17.6
RESHAPE	14	7	24.5	5	5	0.25
CONCURRENT RESHAPE	14	7	27.9	5	5	0.43
CONCURR. MULTIPLE&MIXED	14	6	54.8	5	4*	1.38

*With parameters used in Test-F (Table 4).

Table 3. Results of two test functions

To test concurrency, the same tests given in Tables 1 and 2 were repeated with concurrent and mixed simulatedannealing (Test-A and Test-B). Table 1 shows the computation time is reduced by factors of 7.1 and 6.6 respectively. These speedups are reasonably close to the theoretical maximum of 8 (there are 8 processors).

In addition, experiments were performed on two functions, FUNCTION1 and FUNCTION2, used to compare Cut-or-Combine with Reshape in [2]. FUNCTION1 is a 4-valued 4-variable symmetric function of 176 minterms with a known minimal solution of 6 product terms. It is interesting because it is difficult to minimize by the Cut-or-Combine method. FUNCTION2 is a 4-valued 2-variable function for which Reshape cannot find a minimal solution if it starts in (one of many) specific initial states.

The results are shown in Table 3. Concurrent Reshape produces the same number of product terms as Reshape, 7 and 5 for FUNCTION1 and FUNCTION2, respectively. Concurrent Multiple and Mixed produces the best results in both cases, 6 and 4 product terms for FUNCTION1 and FUNCTION2, respectively. It is interesting that the occasional application of Cut-or-Combine among many applications of Reshape produces a minimal solution, impossible with just Reshape alone.

3.4: Optimum parameters for concurrent and mixed simulated annealing

We consider what effect various parameters have on the performance of the methods discussed above. These parameters are tested with the following values.

a. Maximum frozen factor (3 ; 4 ; 5)
b. Maximum try factor (20 ; 25 ; 29)
c. Maximum valid factor (3 ; 4 ; 5)
d. Initial Temperature (0.50 ; 0.55 ; 0.60 ; 0.62 ; 0.65 ; 0.70 ; 0.75)
e. Cooling Rate (0.90 ; 0.91 ; 0.92 ; 0.93 ; 0.94 ; 0.95 ; 0.97 ; 0.98)
f. Mixture Rate (Cut-or-Combine/Reshape) (10%/90% ; 5%/95% ;4%/96% ; 3%/97% ; 2%/98%)

The use of these parameters is shown in [7]. Briefly, maximum frozen factor represents how many temperature values are allowed at which no new moves are generated before termination. Maximum valid factor, when multiplied by the number of minterms, gives the maximum number of valid moves allowed at each temperature. Maximum try factor, when multiplied by the number maximum number of valid moves, gives the maximum number of attempts at moves that are allowed at each temperature.

We also investigate the effects of temperature dependent mixture rates. Fixing the current temperature, we change the mixture rate as the time changes. This approach did not give better results than the constant mixture rates. There are almost 8,000 combinations of these parameters. However, this is too many to evaluate experimentally. So our approach is to find a near-optimum combination by the following process. At the beginning of our search, all values of "maximum frozen factor" (3; 4; 5) are tested. During these tests, the remaining parameters (b, c, d, e, f) are chosen to be the same as with Reshape in Table 1 (0.7; 4; 5; 0.93; 25), except that a mixture rate of 5%/95% is used. From these tests, the best value (5) is chosen. Keeping this value, the second parameter (maximum try factor) is searched and the best result (25) chosen. The rest of the search is done this way in the order given above for the parameters. After the first pass down to the last parameter, another value is chosen for the third parameter and the search proceeds as before. Notice that more choices are made for the last parameters. In this way, we performed 70 passes for each function across six parameters. Results of these tests are given in Table 4.

	TEST A	TEST B	TEST C	TEST D	TEST E	TEST F	TEST G	TEST H	TEST I	TEST J
Initial Temp	0.7	**0.6**	0.6	0.7	0.65	0.62	0.6	0.65	0.7	0.62
Cooling Rate	0.98	**0.94**	0.94	0.98	0.97	0.91	0.92	0.92	0.91	0.92
Max valid factor	4	**4**	4	4	5	4	4	4	4	4
Max try factor	25	**25**	25	25	25	25	25	25	25	20
Max frozen	5	**5**	5	5	5	5	5	5	5	4
C-or-Cmb/Resh ratio	no mix	**4% / 96%**	no mix	4% / 96%	4% / 96%	4% / 96%	4% / 96%	4% / 96%	4% / 96%	no mix
Function C1	86(62)	**84(93)**	85(74)	84(87)	85(186)	84(61)	86(69)	87(75)	85(71)	85(47)
Function C2	84(67)	**83(96)**	83(70)	83(87)	82(181)	84(59)	83(75)	85(85)	83(74)	84(44)
Function C3	88(80)	**87(97)**	88(75)	88(89)	86(185)	87(70)	88(74)	89(78)	88(71)	89(52)
Function C4	80(74)	**78(65)**	80(64)	80(64)	80(175)	80(59)	80(63)	78(65)	81(54)	82(40)
Function C5	79(66)	**79(92)**	81(73)	79(88)	80(168)	79(59)	80(68)	79(76)	80(60)	81(39)
Function C6	81(62)	**82(96)**	82(69)	83(69)	80(189)	82(68)	82(74)	83(81)	81(62)	83(43)
Function C7	87(69)	**86(108)**	87(75)	68(78)	86(162)	86(74)	86(75)	87(79)	86(71)	87(42)
Function C8	77(55)	**76(76)**	78(69)	77(67)	76(179)	77(58)	77(70)	78(63)	78(53)	77(42)
Function C9	81(75)	**80(90)**	80(91)	81(87)	79(164)	81(67)	82(78)	81(89)	81(68)	83(43)
Function C10	81(65)	**82(86)**	82(82)	83(94)	82(210)	83(65)	83(68)	84(74)	83(82)	84(48)
AVERAGE	82.4(67.5)	**81.7(89.9)**	82.6(74.2)	82.4(81.0)	81.6(179.9)	82.3(64.0)	82.7(71.4)	83.1(76.5)	82.6(66.6)	83.5(44.0)

Table 4. Test comparison

4: Results

Two criteria are used to judge a minimization algorithm, the number of product terms and the computation time. Fig. 3 shows how twelve algorithms compare over a set of ten 4-valued 4-variable functions each having 20 minterms. For example, Reshape in Fig. 3 labels a simulated annealing experiment using Reshape which achieved an average number of product terms of 84.2 with an average computation of 58.0 seconds (the column RSAV in Table 1). If Reshape is replaced by Cut-or-Combine in a single simulated annealing experiment, the result is a large number of product terms (87.0) and a much longer computation time (1990.6 sec.). This is shown as a point labelled Cut-or-Combine in Fig. 3 far away from all other algorithms.

The point labeled A is the experiment discussed in Section 3.3 that uses the same parameters used in regular Reshape except that the "concurrent multiple path" method is implemented. The result is an average of 82.4 product terms with a computation time of 67.5 seconds.

The point labeled D represents an experiment (Table 4) that has the same parameters as A, except that Cut-or-Combine and Reshape moves are mixed in the ratio 4%/96%. The average number of product terms is the same (82.4) as with A, but the computation time (81.0 seconds) is worse. This suggests that different parameters may improve the result. We tested this as follows.

In F (Table 4), the same mixture rate was used as in D but with new parameters (0.62 initial temperature; 0.91 cooling rate). In this case, we see that both the number of product terms and computation time are better than D. This

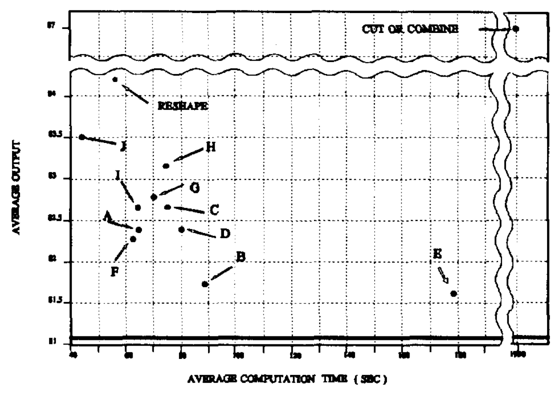

Figure 3. Test comparison.

shows that the implementation of mixed annealing requires different parameters than regular Reshape.

In B (which is the concurrent and mixed method - Table 4), we obtained the best results in terms of product terms with reasonable computation time. The average output is 81.7 product terms on the average with a computation time of 89.9 seconds

The good results shown in B inspired the use of these same parameters in other cases. We considered the application of these parameters to regular Reshape. In C, which is a regular Reshape with multiple paths (Table 4), we used these same parameters but achieved worse results than A. This suggests that the good results produced by concurrent and mixed annealing are not just due to the choice of parameters.

I is the same as F except that the initial temperature was chosen as 0.7 (as in regular Reshape). The computation time is nearly the same, but average number of product terms is worse.

G and H also show how incorrect parameter selection can affect the algorithms performance. In these cases, G is the same as B except that the cooling rate is 0.92 instead of 0.94 and H is the same as G except that the initial temperature is 0.65 instead of 0.60. This indicates that parameter selection is important as a whole.

In applying J (Table 4), we sought a better average output than regular Reshape, but a faster computation time. For this test we used new parameters (0.62 initial temperature; 0.92 cooling rate; 20 maximum try factor - maximum number of tries at each temperature without a move). Indeed, this gave better outputs with a shorter computation time. We see that J fell in the left lower side of the Reshape. As expected, the number of product terms (83.5) was not better than 81.7 for B. But there was a large improvement in computation time to 44.0 seconds compared to 89.9 seconds for B.

Fig. 4 shows a tradeoff between number of product terms and computation time. B seems the best in terms of average computation time and output. That is why we picked it for "Concurrent Multiple and Mixed Annealing" and used for the comparisons below. The solid line at 81.1 indicates the best average output found, from all the tests done (approximately 700) with different parameters. This shows the benefit to parameter and mixture rate selection.

For some of the test results given in Table 4 (A, B, I, J), we also investigated the effects of the increasing number of processors used. Fig. 4 shows the average number of product terms and the computation times, as a function of number of processors used for concurrent multiple and

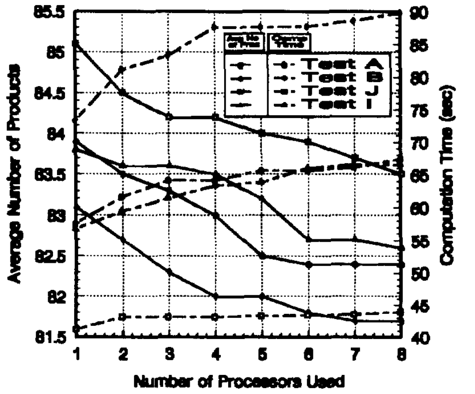

Figure 4. Effects of increasing number of processors.

mixed annealing. J and B seem the best in taking advantage of having more processors.

Fig. 5 shoes a comparison between regular Reshape and Concurrent Multiple and Mixed Annealing for all the functions (C1,..,C10). In this figure, the best results known are also included. For five out of the ten cases, Concurrent Multiple and Mixed Annealing does nearly as well as the best known results, while, for the same functions, Reshape is significantly worse. In all cases, Concurrent Multiple and Mixed Annealing does better than Reshape.

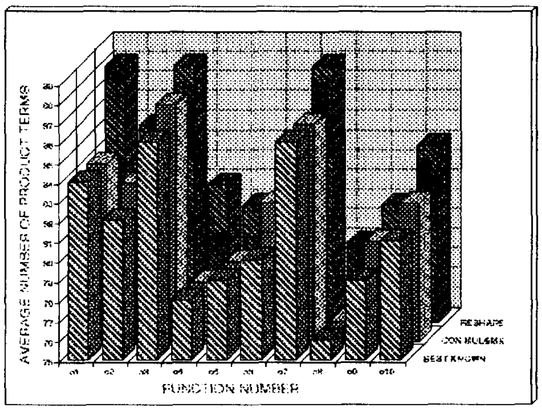

Figure 5. "Reshape", "Concurrent multiple and mixed" and "best known" comparison.

Table 5 shows the comparison with various heuristics described in [1]. The right column gives the rate of improvement in the average number of product terms over penalty for computation time. As a reference, the results of J are included in this table in last row. We see that the outputs of "Concurrent Multiple and Mixed Annealing" are better than the others. The increase in computation time is quite reasonable when we compare the (improvement in average output) / (penalty for computation time) rates for all heuristics.

5: Concluding remarks

An interesting result from our studies is the surprising benefit of mixing two moves, Cut-or-Combine and Reshape. For example, the mixing of a small number of Cut-or-Combine moves (4%) with Reshape moves (96%) allows a minimal solution to be found for a special function, that is impossible if 100% of the moves were Reshape. The benefit of mixing moves was also shown by our experiments on sets of functions where there were a low average number of product terms (e.g. B and D in Fig. 3). As expected, there is an advantage to performing a simulated annealing

HEURISTIC	AVG. PRODUCT TERM	IMPRVM. IN AVG. OUTPUT (%)	AVG. COMP. TIME (sec)	PENALTY IN AVG COMP. TIME (%)	$\frac{\textit{IMPRV. IN AVG OUTPUT (\%)}}{\textit{PENALTY IN COMP. TIME (\%)}}$
POMPER & ARMSTRONG	96.0	0.00 %	8.3	0.00 %	
DUECK & MILLER	94.0	2.08 %	9.7	16 %	0.1300
YANG & WANG	92.0	4.10 %	30.5	266 %	0.0154
CUT-OR-COMBINE	87.0	9.30 %	1990.6	23900 %	0.0004
RESHAPE	83.9	12.60 %	57.6	576 %	0.0218
CONC. MULT & MIXED	81.7	14.90 %	89.3	966 %	0.0154
PARAMETERS GROUP J	83.5	13.00 %	44.0	426 %	0.0300

Table 5. Heuristic comparison.

experiment many times and taking the best result. Doing this on a set of ten functions using eight different experiments yields 82.4 product terms versus 84.2 the expected number for one experiment. This reduction is achieved at a large increase in time, 470.5 secs. for the better result versus 58.0 secs. for the worse.

Running multiple experiments concurrently on independent processors improved the computation time considerably. With eight processors, there is the prospect of a speedup of 8 over a single processor sequentially performing 8 experiments. We found speedups on the order of 7, indicating a diversity in computation times over the 8 experiments (speedups of 8 are achievable only if all experiments require identical computation time.) There is clear advantage in using multiple processors. But, there is also a point of diminishing returns. Our experience suggests that at the eight processors used here, we are beyond that point.

6: Acknowledgements

The authors thank Dr. Parthasarathy Tirumalai for comments that led to improvements in the paper. Also, the two referees provided many good comments.

References

[1] P. Tirumalai and J. T. Butler, "Analysis of minimization algorithms for multiple-valued PLA's," *Proc. of 18th Intl. Symp. on Multiple-Valued Logic*, 1988, pp. 226-236.

[2] G. W. Dueck, R. C. Earle, P. Tirumalai, J. T. Butler, "Multiple-valued programmable logic array minimization by simulated annealing" *Proc. of 22nd Intl. Symp. On Multiple-Valued Logic*, May 1992, pp. 66-74.

[3] S. Kirkpatrick, C. D. Gellat, Jr., and M. P. Vecchi, "Optimization by simulated annealing" *Science*, vol. 220, No. 4598, 13 May 1983, pp. 671-680.

[4] G. W. Dueck, R. C. Earle, P. Tirumalai, J. T. Butler, "Multiple-valued programmable logic array minimization by simulated annealing" Naval Postgraduate School Technical Report NPS-EC-92-004, Feb 1992.

[5] C. Yang and Y. M. Wang, "A neighborhood decoupling algorithm for truncated sum minimization," *Proceedings of the 20th International Symposium on Multiple-Valued Logic*, May 1990, pp. 153-160.

[6] C. Yang and O. Oral, "Experiences of parallel processing with direct cover algorithms for multiple-valued logic minimization," *Proceedings of the 22nd International Symposium on Multiple-Valued Logic*, May 1992, pp. 75-82.

[7] C. Yildirim, "Multiple-valued programmable logic array minimization by concurrent multiple and mixed simulated annealing," M.S. Thesis, Naval Postgraduate School, Monterey, CA, Dec. 1992.